Improving Monte Carlo Localization with Strategic Navigation Policies and Optimal Landmark Placement

Henrique José dos S. Ferreira Júnior[1], **Daniel Ratton Figueiredo**[2]

[1]Escola Politécnica (POLI), Departamento de Engenharia Eletrônica (DEL)
Universidade Federal do Rio de Janeiro (UFRJ) – Brazil

[2]Programa de Engenharia de Sistemas e Computação (PESC)
Universidade Federal do Rio de Janeiro (UFRJ) – Brazil

henriquejsfj@poli.ufrj.br, daniel@cos.ufrj.br

Abstract. *An important problem in robotics is to determine and maintain the position of a robot that moves through a known environment with indistinguishable landmarks. This problem is made difficult due to the inherent noise in robot movement and sensor readings. Monte Carlo Localization (MCL) is a frequently used technique to solve this problem, and its performance intuitively depends on how the robot explores the environment and the position of the landmarks. In this paper, we propose a navigation policy to reduce the number of steps required by the robot to find its location together with the optimal landmark placement for this policy. This proposal is evaluated and compared against other policies using two specific metrics that indicate its superiority.*

1. Introduction

An important problem in robotics is to determine and maintain the position of a robot moving through a previously known environment that has identifiable but indistinguishable landmarks. In this problem, the robot has a complete map of the environment as well as the exact location of landmarks that are scattered throughout the environment. By moving in the environment the robot can detect the presence of a landmark but it cannot discriminate this landmark from any in the set of landmarks present in the environment.

Another challenge in this problem is the inherent noise in the robot movement and landmark detection. In particular, the robot does not move or detect a landmark accurately. For example, when trying to move the north, the robot can remain in the same place, or the robot may fail to detect the presence of a landmark in its current location.

When the robot starts it has no information about its current position in the known map of the environment. The robot must navigate the environment and - with the landmarks detected during it and the environment map - should estimate its position. This is a global localization problem, which differs from the tracking or navigation problem in which the robot starts from a known position on the map and must navigate to some specific destination. The robot's real location is used only to assess the MCL performance.

A technique to solve this global localization problem is Monte Carlo Localization (MCL) originally proposed in 1999 [Fox et al. 1999] but applied to an increasing number of scenarios. Briefly, MCL uses recursive Bayesian inference, when detecting (or not) landmarks, giving more weight to the map positions that match the observations. When moving, MCL uses convolutions, applying the laws of motion to shift the probability

distribution accordingly. As the robot moves and the algorithm is repeated, the belief associated with some locations become larger until a unique location becomes the largest among all others and is then considered the estimated global location of the robot.

1.1. Recent prior works

Our prior work on this topic analyzed the influence of the number of landmarks as well as their position on the effectiveness of MCL when the robot moves randomly through a discrete environment [Ferreira and Figueiredo 2018]. In order to measure the performance of MCL two metrics were proposed, and to solve for the optimal landmark placement, a Simulated Annealing framework was designed and implemented. Our findings indicate that under a random navigation policy, the optimal number of landmarks is 50% of the map locations and their optimal placement is achieved by an uniform random placement.

Our more recent prior work focused on the *active localization problem*, when the robot moves strategically to improve its localization. Assuming the robot is using MCL, we proposed a new navigation policy to greedily reduce the number of possible locations for the robot at each movement [Ferreira and Figueiredo 2019]. This work also proposed a method to compute the best possible sequence of movements for the robot (given a known starting point). The proposed navigation policy and the MCL algorithm was implemented along with a custom simulator specifically designed for this project. The results indicate that the proposed navigation policy can effectively improve localization when compared with random movements. In particular, the performance of the proposed policy on a map with 20% of landmarks placed uniformly randomly is superior to the random policy on a map with 50% of landmarks (the optimal for random navigation).

The simulator along with the proposed policies and the Simulated Annealing framework is publicly available at `https://gitlab.com/henriquejsfj/Monte_Carlo_Localization.git`.

1.2. Current contribution

This current work investigates the influence of the number of landmarks and their placement on the performance of active localization. In particular, assuming MCL and the previously proposed navigation policy, how does the number of landmarks and their placement influence localization performance? In particular, since the navigation policy is strategic, it is not clear that a random placement of landmarks continues to be optimal. Moreover, given a map optimized for an strategic navigation policy, how will a random navigation policy perform on this map?

The above questions are addressed in this work using a simulated annealing approach designed to find the optimal placement of landmarks for a strategic navigation policy. Our previous simulator was adapted and used to establish the quantitative results. Our findings indicate that the optimal map is not random but has many asymmetries, such that a random navigation policy can effectively use the map for localization.

The remainder of this paper is organized as follows. Section 2 briefly presents some related work. Section 3 presents the environment and movement model for the robot as well as the metrics to evaluate the performance of the navigation policies. In section 4 we present the navigation problem and different navigation policies. Section 5 we describe the Simulated Annealing framework designed to find the optimal map.

Section 6 presents and discusses the numerical results of different policies. Finally, a brief conclusion and future work are presented in Section 7.

2. Related Work

Monte Carlo Localization (MCL) [Fox et al. 1999, Thrun et al. 2001] is a widely studied technique for the global robot localization with many improvements like Merge-MCL [Li et al. 2010], and many variations like coping with new sensors such as Bluetooth devices [Hou and Arslan 2017]. However, our work does not focus on improving or adapting MCL, and we assume the classic MCL [Fox et al. 1999] and the classic grid map [Elfes 1987, Milstein 2008]. The focus of this work is to study the influence of a strategic navigation policy on localization performance, a problem known as *active localization* [Fox et al. 1998]. In particular, we study the influence of the number and location of landmarks on MCL under different navigation policies [Ferreira and Figueiredo 2018].

Common solutions to active localization are based on information-theoretic metrics, such as the entropy of the robot's location distribution [Fox et al. 1998, Jensfelt and Kristensen 2001]. Such approaches can be computationally intensive in general environments. In a novel and promising approach, the navigation policy for active localization is learned using reinforcement learning [Gottipati et al. 2019]. Note that active localization is related to the more general problem of simultaneous location and mapping (SLAM) [Rekleitis et al. 2006]. However, our focus is to better understand the classic MCL applied to the classic localization problem.

3. Models and Evaluation Metrics

We consider an environment in the form of a squared two-dimensional grid with wrap-around (i.e., torus) of size $n \times n$. Each location in the environment has exactly four neighboring locations and moving exactly $n + 1$ steps in the same direction returns to the starting point. Each location in the environment corresponds to a possible location for the robot, which in this case has n^2 different locations. Exactly p landmarks are added to the environment, all identical (this means that the robot can know that its current location has a landmark but it does not know what is the location).

Figure 1 illustrates an environment where $n = 10$ and $p = 50$. The presence of a landmark is denoted by a black square and the absence by a white square. However, the performance of MCL in this scenario is symmetric with respect to $p = 50$, such that $p = 49$ has the same expected performance as $p = 51$, and the same happens to $p = 1$ with $p = 99$ (see discussion in [Ferreira and Figueiredo 2018]). Note that if $p > 50$ then the white squares (locations without landmark) can be interpreted as the landmarks. Thus, it is sufficient to vary the number of landmarks between $p = 1$ and $p = 50$ to increase the number of locations that provide more information about the robot's position.

The robot is initially placed in a position chosen uniformly at random within the environment and which is unknown to the robot. We will consider a discrete-time model, where at each step the robot must perform a movement and a reading to identify the presence of a landmark. The movement takes the robot to an adjacent location: up, down, left, or right. Note that the robot is not intended to reach a target, but rather to locate (and remain localized) in the environment. Figure 1 illustrates the position of the robot in an environment with landmarks (black squares) and the possible movements (blue squares).

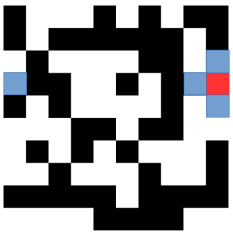

Figure 1. Example of a grid with $p = 50$ with robot in the red square.

To model the noise in the robot motion, we will consider that the robot has a 10% chance of overshooting, i.e. moving an additional position in the same chosen direction, and a 10% chance of undershooting, i.e., staying in the same location. Thus, the attempt to move to the chosen adjacent location occurs satisfactorily with a chance of 80%. It is important to note that the robot is not able to determine if the movement succeed.

To model the noise in landmark detection, a reading tells the robot whether the current location is black (i.e., has a landmark) or white (i.e., does not have a landmark) with a 10% error in each case. That is, the robot correctly detects the presence or absence of a landmark in its current location with a 90% probability.

The variation of MCL that will be used in this work maintains a probability of the robot's position for each location of the map. In particular, $P_x(t)$ is the probability that the robot is in location x of the environment at time t. Note that $P_x(0) = 1/n^2$, because the robot has no information about its initial location. Note that $P_x(t)$ is updated at every time step according to the chosen movement and the observation (or not) of a landmark.

3.1. Evaluation Metrics

Two metrics that were previously proposed will be used to quantify the performance of navigation policies [Ferreira and Figueiredo 2018]. The first metric captures the fraction of movements (steps) where the MCL correctly estimates the position of the robot. The second metric captures the number of movements (steps) required for the robot to find itself correctly for the first time. In the following, we formalize these two metrics.

Let $R(t)$ be the actual location of the robot at time t. Let $H(t) = \{y \mid y = \arg\max_x P_x(t)\}$, where $P_x(t)$ is the probability according to the MCL of the robot being in state x at time t. That is, $H(t)$ is the set of states with the highest probability in time t. Lets define $A(t) = 1$ if $|H(t)| = 1$ and $H(t) = R(t)$, and 0 otherwise. That is, $A(t)$ indicates that the MCL correctly estimates the position of the robot at time t. Let r be the number of steps taken by the robot during an excursion in given environment G. Lets define: $E_G(r) = \sum_{t=1}^{r} A(t)/r$.

The $E_G(r)$ metric captures the intuition that MCL performance is directly proportional to the fraction of time steps in which the robot is correctly located in the environment. $E_G(r)$ is the hit rate of the MCL after r steps, which converges when $r \to \infty$, considering that the environment is finite. However, note that $E_G(r)$ (for fixed r) is a random variable since MCL is random. For a fixed r (e.g. $r = 100$), we can estimate the expected value of $E_G(r)$ by making the sample average of several independent rounds.

Another important metric is the time (measured in number of steps) to the first correct localization. Let F_G be the first instant of time t where $A(t) = 1$ for a given

environment G. Thus, $F_G = \min\{t \mid A(t) = 1\}$.

Notice that F_G is a random variable since it depends on the execution of the algorithm. Again, we report the sample mean of F_G by running the algorithm for many independent rounds. Note that a low value in this metric is important because it indicates that the robot can quickly localize itself in the environment.

3.2. Simulator

A simulator of the model presented to move the robot and detect the presence of landmarks was designed and implemented for this work. Also, the MCL algorithm was implemented to determine the location of the robot as a function of its observations. The choice of designing and implementing our simulator instead of using Gazebo on ROS comes from our simplified model (referred to as map) that is discrete and without borders, and to have a more computationally efficient implementation (since it is simpler).

4. Navigation Policy

A navigation policy determines what movement the robot has to make at each time step. Navigation policy is constructed to achieve a specific goal, usually to minimize a given cost function. Moreover, navigation policies can be static or dynamic in the sense that they can leverage and adapt to the information acquired by the robot. In this work, the goal of the navigation policy is to effectively determine the location of a robot running MCL. This navigation policy is dynamic and will leverage the current belief of the robot to determine its next move. To model a robot going from anyplace to anywhere without a specific objective we used the random movements as an alternative navigation policy. The proposed policy was unique in our attempts to overcome random movements.

The design of a good navigation policy requires determining the expected reading of a movement. Thus, given the current belief of the robot's location and the map, the robot can compute what is the probability of observing a landmark when moving in each possible direction considering both the movement and sensor noises. This probability can then be applied to a function that determines the expected cost for the robot if taking that movement, finally, the robot simply chooses the movement that minimizes it.

An adequate cost function should measure the uncertainty that remains after the movement. Thus, choosing the minimum cost movement leads to greedily choosing a movement that reduces uncertainty. An effective alternative is simply the number of possible locations in $P_x(t)$, namely the number of locations for which $P_x(t) > 0$. Intuitively, this cost function reduces the number of possible locations faster than other cost functions.

In what follows we characterize the behavior of the proposed cost function. Consider a noiseless scenario and let L_t denote the number of possible locations at time t, in particular, $L_t = |\{x|P_x(t) > 0)\}|$. Since we assume a model with no noise, we have that each possible location has probability $\frac{1}{L_t}$, in particular, $P_x(t) = \frac{1}{L_t}, \forall x \in \{x|P_x(t) > 0)\}$. Let p_B and p_W denote the probability of reading black (a landmark) or white (no landmark) after taking a given movement. Let S be a random variable that denotes the sensor reading after the movement, such that $p_b = P(S = b)$ and $p_w = P(S = w)$. Note that $p_b + p_w = 1$. Using this information, the expected value for L_{t+1} for a given movement can be computed as follows:

$$E(L_{t+1}) = p_b * E(L_{t+1}|S = b) + p_w * E(L_{t+1}|S = w)$$

Manipulating this expression for the cost function which captures the expected number of possible locations, we can minimize it as follows (computing its derivative):

$$\frac{d(E(L_{t+1}))}{dE(L_{t+1}|S=b)} = 0 \Longrightarrow \frac{4 * E(L_{t+1}|S=b)}{L_t} = 2 \Longrightarrow E(L_{t+1}|S=b) = \frac{L_t}{2}$$

This result indicates that the expected number of possible locations for L_{t+1} is minimized when a movement divides the prior L_t equally in black and white. Note that this result doesn't depend on the number of landmarks on the map. Note that this cost function is quite similar to the absolute value of the difference between p_b and p_w, namely $|p_b - p_w|$, which is very simple to compute, and is thus used in our navigation policy.

5. Map Optimization

The map in MCL can be interpreted as "a diagrammatic representation of an area showing physical features", where the area is the robot environment and the physical features are the landmarks and obstacles to movement, such as walls. As this work considers a completely symmetric environment (i.e., no walls or obstacles), a map can only be optimized by determining the number and location of the landmarks.

Finding the optimal number of landmarks is easier since we just need to vary the number of landmarks p and measure the performance of MCL. Thus, we generate 100 maps with p landmarks chosen uniformly at random for each map, and first compute E_G (hit rate) and F_G (time to first hit), to then compute its average.

However, finding the optimal placement for the landmarks is not easy, since the number of possible placements is very large. For the map used in this work with $n = 10$ and $p = 50$ there are $\frac{100!}{50!50!} \approx 10^{29}$ possible landmark placements. Since we cannot explore them all, we explore this space through Simulated Annealing (SA) [Skiscim and Golden 1983]. Briefly, SA is an iterative optimization technique that uses a transitioning rule over the state space, and for each iteration chooses the next state with the probability accordingly to the function to be optimized. Usually, Simulated Annealing is used with Boltzmann distribution to calculate each transition probability. In this work, each state is a map with the placement of the landmarks.

The function to be optimized in this work is the hit rate, E_G, and using Boltzmann distribution we have the probability of each map as $\pi_G = e^{\frac{E_G}{T}}/Z$, where $Z = \sum_{G \in \mathcal{G}} e^{\frac{E_G}{T}}$ is the normalizing constant for all the possibles landmarks scattering $\mathcal{G}$ and T the temperature. In this work, we set a constant temperature as $T = 0.01$ and always save the best-visited map instead of decreasing the temperature until convergence.

The transitioning rule of SA leads to a Markov chain model. Our transitioning rule is simple: given a map G, we randomly remove a landmark and add it to a location without a landmark chosen randomly, providing a new map G'. With all the transitions being equiprobable, this makes the Markov chain symmetric that is $p_{G,G'} = p_{G',G}$ for all G, G'. To enforce the Boltzmann distribution over the state space, we use Metropolis-Hasting, a Markov chain transition method. Briefly, Metropolis-Hasting establishes an acceptance probability $a(G, G')$ to transition from state G to G' according to the desired distribution. Due to chain's symmetry and using Boltzmann distribution we have $a(G, G') = \min(1, e^{(E_G - E_{G'})/T})$. Note that if $E_{G'} > E_G$ we will always accept it.

6. Experimental Results

We first evaluate and compare the performances of different navigation policies on maps where the landmarks are randomly placed. Figure 2 shows those performances as a function of the number of landmarks. Note that the hit rate (E_G) of the proposed navigation policy is always higher than that of the random one. Moreover, when the map has 15 landmarks or more, the time to the first hit (F_G) for the proposed policy is also lower.

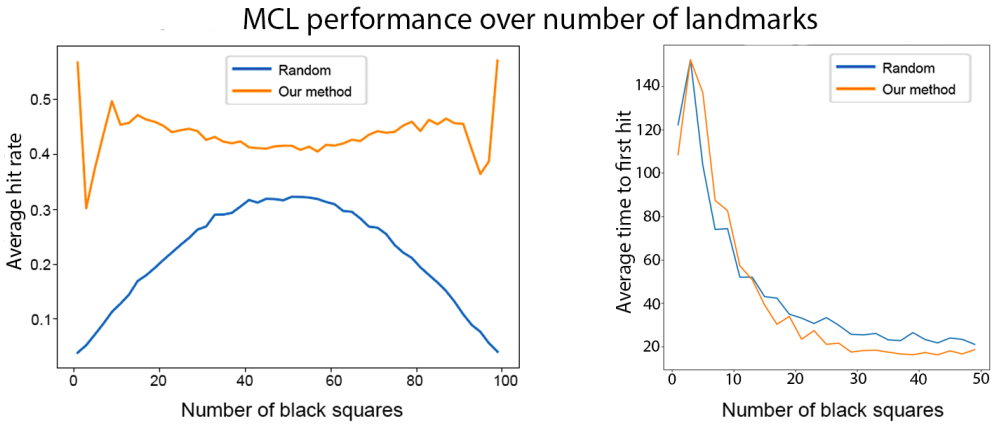

Figure 2. Performances over different number of black squares, performances measured by hit rate (left plot) and measured by time to first hit (right plot).

Interestingly, the performance of the hit rate of the proposed navigation policy is rather unusual: It has a high performance for when the number of landmarks is smaller than 5 and a sharp drop when it reaches around 5 and then improves again.

To better understand the behavior of our navigation policy, we find the best map (landmark placement) for it using the Simulated Annealing (SA) framework, for the different number of landmarks. For each optimal map, we also determine the hit rate for the random navigation policy. However, we also consider the optimal map for the random navigation policy, executing again the SA framework. In this case, we compute the performance of the proposed navigation policy on the optimal map.

Considering the hit rate shown in Figure 2, we start the evaluation of optimal maps with $p = 50$ landmarks, which is where the difference between the two policies is the smallest (under a random map). Figure 3 shows the performance of the two navigation policies as a function of the number of iterations in the SA. The left plot is finding optimal maps for the random policy while the right plot is finding optimal maps for the proposed policy. In both cases, the proposed policy is always superior.

Note that a randomly generated map with $p = 50$ is the best map (landmark placements) for the random policy (a result consistent with prior works [Ferreira and Figueiredo 2018]), as the random policy performance is oscillating around the same value (left plot). Interestingly, the random policy also oscillates around the same value when the map is optimized for the proposed policy (right plot). Thus, optimizing the map for the proposed policy does not affect the performance of the random policy (when $p = 50$).

On the other hand, when optimizing the map for the proposed policy, the SA finds maps where the performance grows to around 0.55 (right plot), showing a significant

improvement over the initial random map, where the performance was around 0.35. Thus, the optimal map for the proposed policy is not a random map.

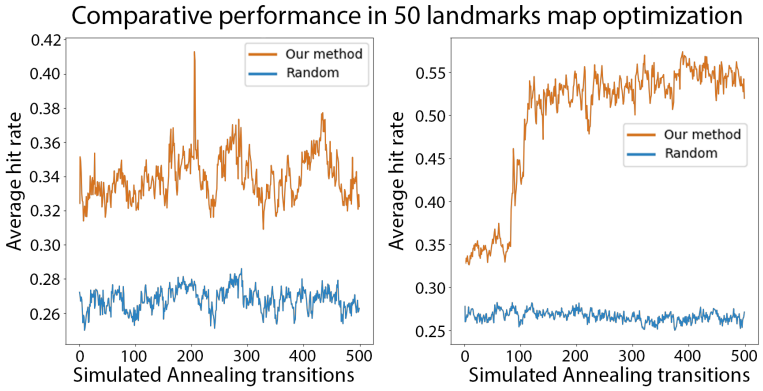

Figure 3. Hit rate as a function of SA iteration for the two policies when finding the optimal map for the random policy (left) and the proposed policy (right).

To explore the sudden drop in performance in the proposed policy shown in Figure 3 for small p, we consider $p = 3$ and find the optimal map for both navigation policies. Figure 4 shows the results as a function of the SA iteration (left and right plots finds optimal maps for random and proposed policies, respectively).

Again, the performance of the random policy remains the same in both scenarios, and is thus not sensitive to changes that are occurring on the map (albeit significantly lower than when $p = 50$). However, the performance of the proposed policy oscillates significantly over the changes in the maps (left plot). What happens is that for three landmarks it is possible to generate really bad maps and also really good ones! This explains the sudden drop in the average hit rate shown in Figure 2 for p around 5.

However, note that when optimizing the map for the proposed policy (right plot) the SA finds maps where the performance is improved and sustained at around 0.56, the same value as with $p = 50$. Thus, under an optimal map, the proposed policy can deliver the same average hit rate with $p = 3$ or $p = 50$.

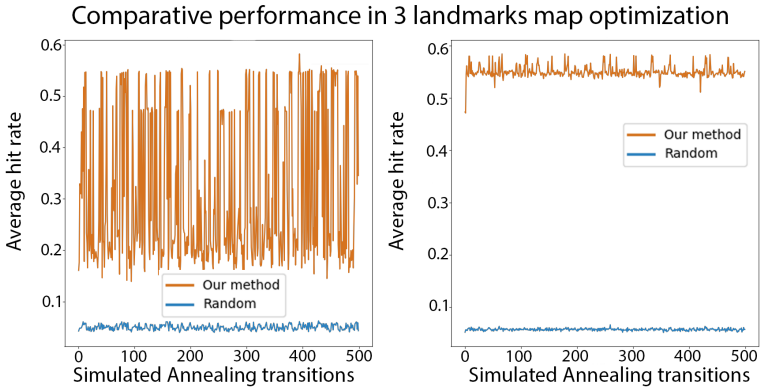

Figure 4. Hit rate as a function of SA iteration for the two policies when finding the optimal map for the random policy (left) and the proposed policy (right).

Beyond the optimal performance for $p = 3$ and $p = 50$, Figure 5 shows the

performance of each navigation policy under its optimal map (found by the SA) as a function of the number of landmarks. Note that the proposed policy has an average hit rate of around 0.56 independently of the number of landmarks, while the performance for random navigation increases with the number of landmarks, to a maximum of around 0.3 for 50 landmarks.

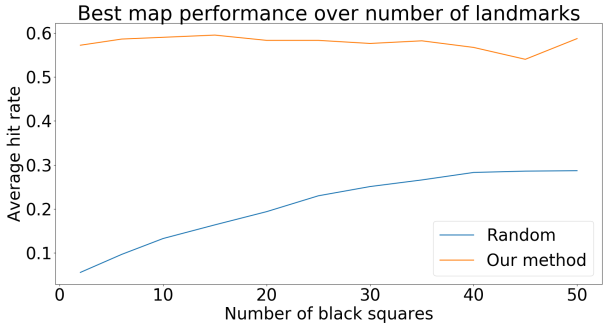

Figure 5. Average hit rate for the optimal map found for each navigation policy as a function of the number of landmarks

Finally, it is interesting to visually inspect the optimal maps for the proposed policy since a random map is clearly not a very good one. Figure 6 shows the initial (random) maps and the optimal maps for a different number of landmarks (i.e. black squares). Note that the optimal map has some symmetries with respect to the initial map. Recall that the proposed policy moves to greedily reduce the difference between black and white squares in its probability distribution. Thus, some level of symmetry is beneficial (with landmarks together, for example). However, too much symmetry would not be effective, since it would be harder (or even impossible) for the robot to find its location.

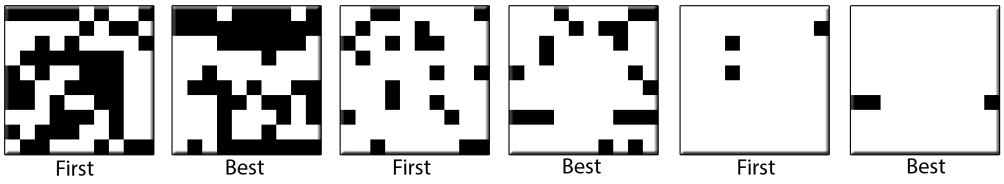

Figure 6. Initial map (chosen at random) and optimal map found by SA for the proposed policy for $p = 50$, $p = 20$ and $p = 3$ (from left to right).

7. Conclusion

Robot localization is a fundamental problem in robotics and is challenging when the robot does not have an external global positioning system and must rely on a map of the environment and landmarks that are not distinguishable. MCL is a fundamental technique that is often used in such scenarios and its performance intuitively depends on the robot's navigation policy and landmark placement.

This work addresses MCL performance under a strategic navigation policy and optimal placement for landmarks with the goal of improving performance. Intuitively, the proposed navigation policy greedily reduces the number of possible locations for the robot at each step. The optimal map offers symmetries and asymmetries that help the robot

find and maintain its location. This map was obtained through a Simulated Annealing optimization framework, starting with a random map.

Extensive simulation results indicate that the proposed navigation is superior to random navigation under a random map and much superior under an optimal map. In particular, under an optimized map, the proposed navigation policy offers a 55% accuracy in the robot's position, independently of the number of landmarks.

An interesting question not explored in this work is the joint task of localization and goal-driven navigation where the robot must move to a specific map location, a problem we leave for future investigation.

References

Elfes, A. (1987). Sonar-based real-world mapping and navigation. *IEEE J. Robotics and Automation*, 3(3):249–265.

Ferreira, Henrique, J. and Figueiredo, D. (2018). Influence of location and number of landmarks on the monte carlo localization problem. In *XV Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*.

Ferreira, Henrique, J. and Figueiredo, D. (2019). Improving monte carlo localization performance using strategic navigation policies. In *XVI Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*.

Fox, D., Burgard, W., Dellaert, F., and Thrun, S. (1999). Monte carlo localization: Efficient position estimation for mobile robots. *AAAI/IAAI*, 1999(343-349):2–2.

Fox, D., Burgard, W., and Thrun, S. (1998). Active markov localization for mobile robots. *Robotics Auton. Syst.*, 25(3-4):195–207.

Gottipati, S. K., Seo, K., Bhatt, D., Mai, V., Murthy, K., and Paull, L. (2019). Deep active localization. *IEEE Robotics and Automation Letters*, 4(4):4394–4401.

Hou, X. and Arslan, T. (2017). Monte carlo localization algorithm for indoor positioning using bluetooth low energy devices. In *Int. Conf. on Localization and GNSS (ICL-GNSS)*, pages 1–6.

Jensfelt, P. and Kristensen, S. (2001). Active global localization for a mobile robot using multiple hypothesis tracking. *IEEE Trans. Robotics and Automation*, 17(5):748–760.

Li, T., Sun, S., and Duan, J. (2010). Monte carlo localization for mobile robot using adaptive particle merging and splitting technique. In *IEEE Int. Conf. on Information and Automation (ICIA)*, pages 1913–1918.

Milstein, A. (2008). Occupancy grid maps for localization and mapping. In *Motion Planning*. InTech.

Rekleitis, I. M., Meger, D., and Dudek, G. (2006). Simultaneous planning, localization, and mapping in a camera sensor network. *Robotics Auton. Syst.*, 54(11):921–932.

Skiscim, C. C. and Golden, B. L. (1983). Optimization by simulated annealing: A preliminary computational study for the TSP. In *Winter Simulation Conference*, pages 523–535.

Thrun, S., Fox, D., Burgard, W., and Dellaert, F. (2001). Robust monte carlo localization for mobile robots. *Artif. Intell.*, 128(1-2):99–141.